Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Selecting features for object detection using an AdaBoost-compatible evaluation function

Luka Fürst *, Sanja Fidler, Aleš Leonardis

University of Ljubljana, Faculty of Computer and Information Science, Visual Cognitive Systems Laboratory, Tržaška 25, SI-1001 Ljubljana, Slovenia

A R T I C L E   I N F O

Article history:
Received 19 February 2007
Received in revised form 11 January 2008
Available online 14 April 2008

Communicated by L. Heutte

Keywords:
Feature selection
AdaBoost
Object detection

A B S T R A C T

This paper addresses the problem of selecting features in a visual object detection setup where a detection algorithm is applied to an input image represented by a set of features. The set of features to be employed in the test stage is prepared in two training-stage steps. In the first step, a feature extraction algorithm produces a (possibly large) initial set of features. In the second step, on which this paper focuses, the initial set is reduced using a selection procedure. The proposed selection procedure is based on a novel evaluation function that measures the utility of individual features for a certain detection task. Owing to its design, the evaluation function can be seamlessly embedded into an AdaBoost selection framework. The developed selection procedure is integrated with state-of-the-art feature extraction and object detection methods. The presented system was tested on five challenging detection setups. In three of them, a fairly high detection accuracy was effected by as few as six features selected out of several hundred initial candidates.

1. Introduction

Feature selection, a process that follows feature extraction in a typical visual classification or detection setup, serves multiple purposes simultaneously. Obviously, a smaller set of features implies less computational effort in the test stage. A successful selection method might also improve classification or detection accuracy and enhance robustness.

Viola and Jones (2004) demonstrate that a powerful selection method makes it possible to use a fairly simple feature extractor. In their method, a vast number of features is extracted using four elementary rules. Most of these features seem to be devoid of any discriminative power. However, by using an effective selection strategy, Viola and Jones manage to find a small set of features such that the detection algorithm in the test stage achieves a remarkable detection accuracy with real-time performance.

For the purpose of explanation, we divide the feature selection module into the *feature evaluation function*, which assigns a score to each extracted feature, and the *selection procedure*, which makes use of the evaluation function to find a favorable set of features. Although in our method the evaluation function is actually embedded in the selection procedure, the paper deals with these two entities separately.

In this paper, feature selection is considered in the context of visual object detection. However, the proposed selection method is based on techniques that were initially designed for (binary) *classification* rather than detection setups. For this reason, we will now briefly delineate the problem of feature selection in a classification setup.

The goal in binary classification setups is to classify individual images into two classes, often termed 'positive' and 'negative' class. The feature evaluation function can be formulated in a variety of ways. A common solution is to reward (i.e., assign a high score to) features that occur frequently in positive but seldom in negative images. Mutual information, correlation, and likelihood ratio are all derived from this general idea. Dorkó and Schmid (2003) compare the performance of mutual information and likelihood ratio for object classification. Epshtein and Ullman (2005) construct a hierarchy of fragments for visual classification and employ mutual information to select the most informative fragments at each level. The evaluation function designed by Li et al. (2006) rewards a feature F extracted from a class-C image if F is, given a distance metric, close to many other features extracted from class-C images and far from most features extracted from images that do not belong to class C.

In object *detection*, where the goal is to find all instances of a particular visual category (e.g., cars) in individual images, such feature evaluation functions can be used if the given detection problem is transformed into a classification problem. Viola and Jones (2004) convert the problem of detecting faces in images into that of classifying face-sized image windows. A window belongs to the positive class if and only if it contains a face. Their evaluation function rewards features that assume generally high values in

* Corresponding author. Fax: +386 1 4264 647.
E-mail address: luka.fuerst@fri.uni-lj.si (L. Fürst).

positive windows and generally low values in negative windows, or vice versa. Shotton et al. (2005) detect objects in a given image via classifying individual locations in the image. A location is regarded as positive if and only if it is sufficiently close to the centroid of a displayed object.

We perform a detection-to-classification transformation only in the feature selection stage. The training image set consists of positive images, which contain examples of objects, and negative images, which do not. For each feature, a classifier is constructed that classifies individual training images as positive or negative using a detection-based criterion. A feature is then assigned a high score if the classifier associated with it classifies most training images correctly. The feature evaluation function defined in this way is incorporated into a selection procedure based on the AdaBoost algorithm (Freund and Schapire, 1997; Viola and Jones, 2004).

Our approach is partially inspired by the object detection method of Opelt et al. (2006). The authors originate from premises similar to ours, but they design a different feature evaluation function. In contrast to Opelt et al., we also consider the influence of the number of selected features on detection accuracy. In particular, we show that the number of features required to achieve favorable detection results may be very low.

This paper brings forth two contributions. First, we devise a novel formal criterion for evaluating features in an object detection setup and integrate it into an AdaBoost-based selection strategy. The presented selection scheme is combined with a detection procedure based on Leibe et al. (2004) and by the feature extraction algorithm of Fidler et al. (2006). The unification of the powerful extraction, selection, and detection mechanisms is our second contribution.

The rest of the paper is structured as follows. Section 2 derives the feature evaluation function and presents the selection procedure. Section 3 describes the detection module. Section 4 outlines the feature extraction module. In Section 5, the method is experimentally evaluated. Section 6 brings the paper to a conclusion.

2. Feature selection

In this section, we derive the feature evaluation function (Section 2.2) and describe its incorporation into the AdaBoost selection procedure (Section 2.3). Prior to that, in Section 2.1, the selection module is presented in terms of its interface and its relationship with the detection module.

Fig. 1 displays a schematic overview of the entire selection module.

2.1. The selection module in terms of its interface and its relationship with the detection module

The following items constitute the input to the selection procedure:

- An initial feature set, $\mathscr{F} = \{F_i\}_i$.
- A training image set, $\mathscr{I} = \mathscr{I}_+ \cup \mathscr{I}_-$.
- The desired number of selected features, T.

The image set should contain both positive images ($\mathscr{I}_+$) and negative images ($\mathscr{I}_-$). Each positive image should display an object of the target category and should be annotated by the object's bounding box. Negative images may display anything except any objects of the target category.

The selection procedure returns the set of selected features, $\{F^{[1]}, \ldots, F^{[T]}\}$, for a given T. The number T should be given in advance. This paper does not deal with the problem of seeking the optimal number of selected features.

The selection procedure uses the detection module and therefore makes certain assumptions regarding its functioning. Given a set of features $\mathscr{G} \subseteq \mathscr{F}$ and an image I, the detection algorithm (a part of the detection module) is expected to produce object detection hypotheses for I using only features in $\mathscr{G}$. Each detection hypothesis H should be accompanied by a non-negative real number $s(H)$, called 'strength' or '(confidence) score', which indicates

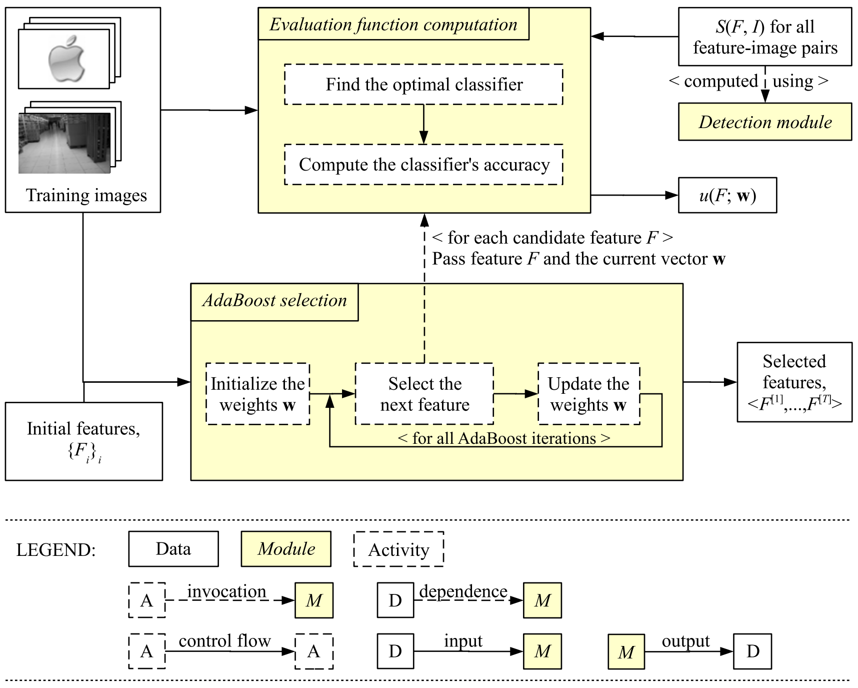

Evaluation function computation
Find the optimal classifier
Compute the classifier's accuracy
S(F, I) for all feature-image pairs
< computed using >
Detection module
u(F; w)
Training images
< for each candidate feature F >
Pass feature F and the current vector w
AdaBoost selection
Initialize the weights w
Select the next feature
Update the weights w
< for all AdaBoost iterations >
Selected features, <F[1],...,F[T]>
Initial features, {F_i}_i
LEGEND:
Data
Module
Activity
A invocation M
D dependence M
A control flow A
D input M
M output D

Fig. 1. A schematic depiction of the feature selection module. In each iteration, the AdaBoost selection procedure invokes the evaluation function computation unit for each candidate feature, passing the current weight vector to the unit. The values $S(F, I)$ are calculated for each feature–image pair in advance, using the detection module. See the subsequent text for explanation.

the detection algorithm's degree of belief that hypothesis H correctly describes the position and size of an object in I. Whether H is indeed correct should be determined by a procedure independent of the detection algorithm. Any detection method that fulfills these minimum requirements may be used in conjunction with the proposed selection procedure.

2.2. The feature evaluation function

This section gradually derives the feature evaluation function, $u : \mathscr{F} \to \mathbb{R}_0^+$, which assigns score to individual features from $\mathscr{F}$. To begin with, let us define a threshold-based binary classifier that classifies training images according to the following rule:

$$d_{F,\theta}(I) = \begin{cases} 1 & \text{if } S(F,I) \geqslant \theta; \\ 0 & \text{if } S(F,I) < \theta. \end{cases} \tag{1}$$

The function $S : \mathscr{F} \times \mathscr{I} \to \mathbb{R}$ will be defined later; for the time being, no definite meaning is to be associated with it. Classifier $d_{F,\theta}$ classifies image I as positive ($d_{F,\theta}(I) = 1$) if the value of function S for feature F and image I is greater than or equal to threshold θ. Otherwise, the image is classified as negative.

Let y_I represent the true class of training image I: $y_I = 1$ if $I \in \mathscr{I}_+$, and $y_I = 0$ if $I \in \mathscr{I}_-$. The expression $|d_{F,\theta}(I) - y_I|$ thus equals 0 for all correctly classified images and 1 for all misclassified images. The *error rate* of classifier $d_{F,\theta}$ is calculated as a weighted sum of such expressions over the entire training image set:

$$\epsilon(F, \theta; \mathbf{w}) = \sum_{I \in \mathscr{I}} w_I |d_{F,\theta}(I) - y_I|. \tag{2}$$

The weight vector $\mathbf{w} = (w_I)_{I \in \mathscr{I}}$ is a variable manipulated by the AdaBoost algorithm (Section 2.3). The vector's initial value is $\mathbf{w}^{[1]} = (w_I^{[1]})_{I \in \mathscr{I}}$, where

$$w_I^{[1]} = \begin{cases} 1/(2|\mathscr{I}_+|) & \text{if } I \in \mathscr{I}_+; \\ 1/(2|\mathscr{I}_-|) & \text{if } I \in \mathscr{I}_-. \end{cases} \tag{3}$$

By this definition, the positive and the negative images initially have the same collective weight: $\sum_{I \in \mathscr{I}_+} w_I^{[1]} = \sum_{I \in \mathscr{I}_-} w_I^{[1]} = 1/2$ (Freund and Schapire, 1997). After initialization, AdaBoost iteratively modifies vector $\mathbf{w}$.

As shown in Fig. 1, the evaluation function is designed to act as a subroutine of the AdaBoost selection procedure. However, the evaluation function may also be used directly to select features. In this case, $\mathbf{w}$ has to be a constant rather than a variable; we use $\mathbf{w} = \mathbf{w}^{[1]}$. The subsequent equations involve $\mathbf{w}$ as a parameter, so they apply to both selection approaches.

Given a feature F and a weight vector $\mathbf{w}$, the error rate of classifier $d_{F,\theta}$ depends solely on threshold θ. We will be interested only in the *optimal threshold*, which is the value of θ that leads to the minimum error rate:[1]

$$\theta^*(F; \mathbf{w}) = \arg\min_{\theta} \epsilon(F, \theta; \mathbf{w}). \tag{4}$$

To find the optimal threshold for the given F and $\mathbf{w}$, at most $|\mathscr{I}| + 1$ distinct candidate θ values have to be checked (plugged into (2)). To see this, let $\langle S_1, \ldots, S_{|\mathscr{I}|} \rangle$ represent the ascendingly sorted sequence of values $S(F,I)_{I \in \mathscr{I}}$ for the given F. The intervals $(-\infty, S_1], (S_1, S_2], \ldots, (S_{|\mathscr{I}|-1}, S_{|\mathscr{I}|}], (S_{|\mathscr{I}|}, \infty)$ collectively cover the entire axis $\mathbb{R}$, so the optimal threshold must reside in one of them. Consider now the interval $(S_i, S_{i+1}]$ for some i. All thresholds $\theta \in (S_i, S_{i+1}]$ make the classifier $d_{F,\theta}$ produce the same output on $\mathscr{I}$: those images for which $S(F,I) \geqslant S_{i+1}$ are classified as positive, and all others as negative, regardless of the exact value of θ within this interval. Therefore, all $\theta \in (S_i, S_{i+1}]$ lead to the same error rate. This reasoning applies to all considered intervals. Each interval is a set of thresholds that lead to the same classification error rate. Therefore, to find the optimal threshold, it suffices to check one arbitrary threshold per interval. Since there are at most $|\mathscr{I}| + 1$ intervals, at most $|\mathscr{I}| + 1$ candidates for the optimal threshold have to be checked.

[1] To speak of 'the' optimal threshold is somewhat misleading, since there may be several θ values leading to the minimum error rate. However, all optimal thresholds will be considered equivalent. 'The optimal threshold' will thus represent any optimal threshold for the given classification setup.

The *optimal classifier* for feature F and vector $\mathbf{w}$ classifies training images using the optimal threshold for F and $\mathbf{w}$: $d^*_{F;\mathbf{w}} \equiv d_{F,\theta^*(F;\mathbf{w})}$. The *feature evaluation function* is defined as the accuracy of the optimal classifier over $\mathscr{I}$:

$$u(F; \mathbf{w}) = 1 - \epsilon(F, \theta^*(F; \mathbf{w}); \mathbf{w}). \tag{5}$$

Function u will be fully determined once the function $S : \mathscr{F} \times \mathscr{I} \to \mathbb{R}$ is defined. Let us first describe how the distribution of the $S(F,I)$ values over $I \in \mathscr{I}$ for a fixed F and for $\mathbf{w} = \mathbf{w}^{[1]}$ affects the evaluation function's value.

Consider the situation in which the distribution $(S(F,I))_{I \in \mathscr{I}}$ consistently assumes relatively high values for positive images and relatively low values for negative images (Fig. 2). In this case, the optimal threshold will lie somewhere between these 'high' and 'low' values, and the optimal classifier will correctly classify most of the training images. As a consequence, feature F will be assigned a high evaluation score $u(F; \mathbf{w}^{[1]})$. If, however, the distribution of $S(F,I)$ over the training image set were uniform or random, then even the optimal classifier would be unable to achieve a low error rate; regardless of the threshold value, a considerable percentage of training images would always be misclassified. Feature F would hence be assigned a low evaluation score.

The evaluation function should reward features that are deemed to be 'potentially useful' for detecting objects of the given category. We consider a feature F to be 'potentially useful' if it possesses the following properties:

Distinctiveness. F occurs frequently in positive and rarely in negative training images.

Predictiveness. The detection algorithm, if limited to the use of feature F, often correctly predicts the object's position in positive images.

A feature F that is both distinctive and predictive should thus be assigned a high score $u(F; \mathbf{w}^{[1]})$, and vice versa. The score depends on the distribution $(S(F,I))_{I \in \mathscr{I}}$, which should therefore be favorable (high for $I \in \mathscr{I}_+$, low for $I \in \mathscr{I}_-$) for a 'potentially useful' feature F. As will be shown shortly, the following definition of function S makes the evaluation function model the 'potential usefulness' criterion in the desired way:

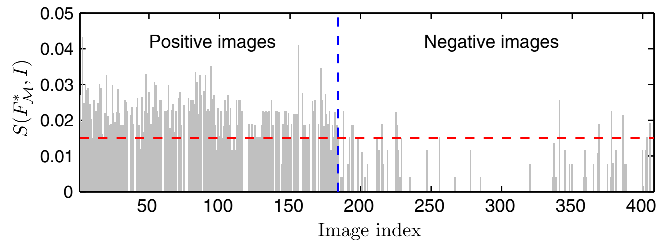


Fig. 2. The distribution $(S(F^*_{\mathscr{M}}, I))_{I \in \mathscr{M}}$, where $\mathscr{M}$ denotes the training image set for the ETHZ mugs dataset (details in Section 5) and $F^*_{\mathscr{M}}$ represents the highest-scored feature obtained for $\mathscr{M}$. The horizontal dashed line visualizes the optimal threshold for $\mathbf{w} = \mathbf{w}^{[1]}$. The optimal classifier misclassifies 26 (out of 184) positive images and 16 (out of 224) negative images. For 20 positive images, $S(F^*_{\mathscr{M}}, I)$ equals 0. These images would be misclassified for any positive threshold.

- For a positive training image I_+:

$$S(F, I_+) = \begin{cases} s(H^*(I_+|F)) & \text{if } H^*(I_+|F) \quad \text{is correct;} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

- For a negative training image I_-:

$$S(F, I_-) = s(H^*(I_-|F)). \tag{7}$$

$H^*(I|F)$ denotes the strongest detection hypothesis obtained by applying the detection algorithm to image I, with the additional requirement that the detection algorithm use only feature F to build hypotheses. $s(H)$ denotes the strength of a hypothesis H. The correctness of each hypothesis is determined as described in Section 5.3.

If a given feature F is both distinctive and predictive, $S(F, I)$ is, according to (6) and (7), generally high for positive and low for negative images, which implies a high value of $u(F; \boldsymbol{w}^{[1]})$. If feature F is indistinctive, the distribution of $S(F, I)$ values over the image set is more or less random. If F is non-predictive, $S(F, I)$ is zero for many positive images. In both cases, F is assigned a low evaluation score. To summarize, the evaluation function rewards a feature if and only if it is 'potentially useful.' Our definition of functions S and u is therefore justified.

2.3. The selection procedure

Given a desired number of selected features, T, the simplest selection procedure returns the T features with the highest values of $u(F; \boldsymbol{w}^{[1]})$. However, such a selection approach tends to produce redundant selection sets, since features with similar evaluation scores may exhibit a high degree of *interdependence*. This phenomenon is discussed in Cover (1974) and Peng et al. (2005).

Owing to its design, which was motivated by Viola and Jones (2004), the evaluation function can be straightforwardly incorporated into an AdaBoost selection framework, which deals with feature interdependence in an implicit fashion. Although AdaBoost was developed as a method to enhance classification accuracy by combining simple classifiers (Freund and Schapire, 1997), it has recently been adapted as a feature selection technique in many classification and detection setups (Viola and Jones, 2004; Torralba et al., 2004; Shotton et al., 2005; Zhang et al., 2005; Opelt et al., 2006; Li et al., 2006).

The (simplified) AdaBoost procedure, shown as Algorithm 1, iteratively selects features and updates the image weight vector $\boldsymbol{w}$. In each iteration, AdaBoost selects the feature ($F^{[t]}$) that achieves the highest score $u(F; \boldsymbol{w})$ with respect to the current value of $\boldsymbol{w}$. After that, AdaBoost decreases the weights of the images that were correctly classified by the optimal classifier ($d^{[t]}$) for the selected feature $F^{[t]}$. The relative importance of the images misclassified by $d^{[t]}$ thus increases. As a result, the next iteration is likely to select a feature such that the associated optimal classifier, $d^{[t+1]}$, performs well on the images that were misclassified by $d^{[t]}$. The described weight-adjusting mechanism thus encourages the complementarity between successive selections, which leads to a lower degree of redundancy in the final set of selected features.

3. The detection module

The implemented detection module is based on the method of Leibe et al. (2004) (presented in detail in Leibe (2004)), which is fairly robust to noise, occlusion, and intra-class variation. The underlying star-shaped model, ISM (Implicit Shape Model), has been successfully applied to various object detection and classification setups (Fritz et al., 2005; Mikolajczyk et al., 2006; Thomas et al., 2006).

Section 3.1 presents a general idea of the algorithm to learn ISM. Section 3.2 describes the most basic detection algorithm, which assumes that the scale (i.e., size) of all test objects is fixed and known in advance. Section 3.3 presents an algorithm that often substantially improves the detection accuracy of the basic approach. Section 3.4 outlines how the method can be adapted to the multiscale detection scenario, which requires the determination of the unknown size, in addition to the unknown location, of each test object. Section 3.5 contains some concluding remarks.

3.1. Learning

The learning algorithm accepts a set of local features $\mathscr{G}_0$ and a set of training images represented by these features. Each image should display an object with a known bounding box. In each image, the center of the object's bounding box is regarded as the reference point, the 'object center.' For each feature G_i, the learning algorithm nonparametrically estimates the probability distribution $p(\boldsymbol{c}|G_i)$ using the training images. For each pair $(\boldsymbol{c}, G_i)$, $p(\boldsymbol{c}|G_i)$ signifies the probability that the object center is located at position $\boldsymbol{c} + \boldsymbol{d}$ if feature G_i occurs at an arbitrary position $\boldsymbol{d}$.

Algorithm 1. The simplified AdaBoost algorithm for feature selection

```
Require:
    (a) A training image set $\mathscr{I}$ with class labels $y_I$.
    (b) The initial feature set, $\mathscr{F}$.
    (c) The desired number of features (iterations), $T$.
Ensure: The sequence of selected features, $\langle F^{[1]}, \ldots, F^{[T]} \rangle$.
Algorithm:
    1: $\mathscr{F}' := \mathscr{F}$;
    2: Compute the initial weights, $\boldsymbol{w}^{[1]} = \{w_I^{[1]}\}_I$, using (3).
    3: for $t = 1$ to $T$ do
    4: Normalize the image weights $\boldsymbol{w}^{[t]} = \{w_I^{[t]}\}_I$:
         $w_I^{[t]} := \frac{w_I^{[t]}}{\sum_{J \in \mathscr{I}} w_J^{[t]}}$   for all $I \in \mathscr{I}$.   (8)
    5: Select the feature having the highest evaluation score
       with respect to the current weights:
         $F^{[t]} = \arg\max_{F \in \mathscr{F}'} u(F; \boldsymbol{w}^{[t]})$.   (9)
       Let $\theta^{[t]} = \theta^*(F^{[t]}; \boldsymbol{w}^{[t]}), d^{[t]} = d_{F^{[t]}, \theta^{[t]}}$, and $\epsilon^{[t]} = \epsilon(F^{[t]}, \theta^{[t]}; \boldsymbol{w}^{[t]}) = 1 - u(F^{[t]}; \boldsymbol{w}^{[t]})$ denote the optimal threshold,
       the optimal classifier, and the optimal classifier's error
       rate, respectively, for the selected feature.
    6: $\mathscr{F}' := \mathscr{F}' \setminus \{F^{[t]}\}$.
    7: Update the weights:
         $w_I^{[t+1]} = \begin{cases} \frac{\epsilon^{[t]}}{1-\epsilon^{[t]}} w_I^{[t]} & \text{if } d^{[t]}(I) = y_I \\ w_I^{[t]} & \text{otherwise.} \end{cases}$   (10)
    8: end for
```

3.2. The basic single-scale detection algorithm

The detection algorithm accepts an input image and a feature set $\mathscr{G} = \{G_1, \ldots, G_m\} \subseteq \mathscr{G}_0$. The image has to be represented by features from $\mathscr{G}$. The visual elements extracted from the image (a *visual element* shall refer to an image entity that can be directly matched against individual features) *vote* for various object center hypotheses. According to Leibe (2004), the cumulative voting score for a hypothesized object center location $\boldsymbol{c}$ amounts to

$$s_0(\boldsymbol{c}) = \sum_k \sum_{i=1}^{m} p(\boldsymbol{c}|G_i, \boldsymbol{p}_k), P(G_i|E_k), \tag{11}$$

where E_k denotes a visual element extracted at position $\boldsymbol{p}_k$. The probability $p(\boldsymbol{c}|G_i, \boldsymbol{p}_k)$ is by the basic ISM assumption equal to the learned probability $p(\boldsymbol{c} - \boldsymbol{p}_k|G_i)$.

The final set of hypotheses is composed of stable local maxima of function $s : \mathbb{R}^2 \to \mathbb{R}$, which may be obtained by smoothing the function s_0 or by the more complex Mean-Shift algorithm. The confidence score, or *strength*, of a final hypothesis $H \equiv \boldsymbol{c}$ (denoted by $s(H)$) is computed from the values of s_0 in the neighborhood of point $\boldsymbol{c}$ as determined by the smoothing kernel.

3.3. The hypothesis attenuation algorithm

This algorithm is able to improve the detection accuracy if (i) in most object-containing test images, the strongest hypothesis correctly detects one of the displayed objects, but (ii) the overall detection accuracy is negatively affected by the presence of globally strong incorrect hypotheses, which might not be the strongest within their source images, but are still stronger than many correct hypotheses in the entire image set.

According to the voting scheme, a single visual element might contribute score to several hypotheses in the image. In the first iteration of the attenuation algorithm, all visual elements E^* that voted for the strongest hypothesis (H^*) are *uniquely* assigned to H^*. The strength of H^* does not change, but each of the remaining hypotheses loses the score that it received from elements E^*. As a result, a hypothesis $H \neq H^*$ that originally drew much of its strength from elements E^* becomes substantially weaker. Hypothesis H^* and elements E^* are then eliminated from further consideration. In the next iteration, the algorithm again finds the strongest hypothesis and uniquely assigns all contributing elements to it. The process repeats until all visual elements have been uniquely assigned. In practice, the conditions from the previous paragraph are often met, so this simple algorithm greatly increases detection accuracy.

The described algorithm was designed by Leonardis et al. (1995) as an approach to solving the general model selection problem. The algorithm, which is based on the MDL (Minimum Description Length) principle, has been thereafter applied to various problem domains that involve model selection. Leibe et al. (2004) successfully applied it to the object detection domain.

For the rationale and the theoretical foundations of the described algorithm, the reader is referred to the aforementioned papers and to Leibe (2004).

3.4. Detecting objects at multiple scales

The presented method can be straightforwardly extended to the multiscale scenario. The learning algorithm requires no changes; it still builds a single-scale representation. The basic multiscale detection algorithm constructs a scale-space pyramid for each input image and performs the voting procedure for each pyramid level separately. By contrast, the attenuation algorithm is adapted to operate over the whole pyramid.

3.5. Concluding remarks

The detection module is used in two different contexts. In the test stage, the detection algorithm employs the selected features to detect objects in test images. The detection algorithm is also invoked in the training stage, when computing function S (Eqs. (6) and (7)). To reduce the processing time, a simplified version of the detection algorithm is used in this latter case. This simplified algorithm analyzes each input image only at its original scale and does not include the hypothesis attenuation algorithm.

To determine the correctness of a detection hypothesis (Section 5.3), a bounding box has to be predicted for each hypothesized object. The center of each hypothesized bounding box is set to the predicted object center location. In the single-scale version, the size of each bounding box is set to the average training object size. To predict the size of the bounding box for a hypothesis H in the multiscale version, the average training size is divided by the relative scale at which H has been detected.

4. Parts composed of parts as the underlying features

The feature extraction stage is instantiated by the method developed by Fidler et al. (2006), which builds a hierarchy of 'parts composed of parts.' Layer 1 of the hierarchy consists of atomic parts, each of which takes the form of an oriented Gabor filter kernel. Each part of a general layer $l > 1$ is a loose geometric composition of one or more parts of layer $l - 1$ and is robustly described by the locations and orientations of its constituents (subparts) relative to the constituent in the center of the composition.

Given a set of training images, the hierarchy of parts for the target visual category is learned in a layer-by-layer fashion by a statistically driven procedure. The constructed parts of a chosen layer L form the set of extracted features, which subsequently enters our selection procedure.

To represent a (novel) image by layer-L parts, the image is first represented by layer-1 parts, then by layer-2 parts, etc. Representing an image by layer-1 parts includes convolving the image with the layer-1 filter collection, storing the maximum-response filter index for each pixel, and performing an information-compaction step. Since parts of a layer $l > 1$ are composed of parts of layer $l - 1$, the layer-l representation can be obtained from the layer-$(l - 1)$ representation via a straightforward matching procedure.

For more information, the reader is referred to Fidler et al. (2006) and to Fidler and Leonardis (2007).

5. Experimental evaluation

The proposed approach was evaluated on the following test image datasets:

- UIUC cars (Agarwal et al., 2004)[2]: Comprises 108 images displaying 139 objects of interest in total.
- INRIA horses[3]: 170 images; 180 objects.
- ETHZ[4] Apple® logos: 40 images; 44 objects.
- ETHZ bottles: 48 images; 55 objects.
- ETHZ mugs: 48 images; 66 objects.

For each dataset, a separate detection setup was constructed.

The INRIA and ETHZ datasets (Ferrari et al., 2006) exhibit a much greater view and scale variance than the UIUC dataset. On the other hand, UIUC includes both left-to-right and right-to-left cars, whereas the other datasets involve only one (broad) view direction.

Sections 5.1 and 5.2 provide some implementation details on the training and the test stage, respectively. Sections 5.3 and 5.4 present the employed evaluation criteria and detection accuracy measure. Section 5.5 presents and discusses experimental results.

[2] http://l2r.cs.uiuc.edu/~cogcomp/Data/Car.

[3] http://pascal.inrialpes.fr/data/horses.

[4] http://www.vision.ee.ethz.ch/~ferrari.

Table 1
Selected implementation details

Setup	Number of training images (pos./neg.)	Average training object size (width × height)	Scale range for test images[a]
Cars	100 + 550/500[b]	218 × 71	−5 to +1
Horses	328/182	141 × 109	−10 to +1
Apple logos	54/420	98 × 120	−8 to +3
Bottles	66/366	58 × 219	−7 to +3
Mugs	184/224	149 × 137	−7 to +3

[a] Designator r corresponds to a relative scale of $2^{r/4}$. The range 'a to b' thus implies that the method is potentially able to detect objects of size between $2^{-b/4}R_0$ and $2^{-a/4}R_0$, where $R_0 = (\text{width}_0, \text{height}_0)$ denotes the average training object size.
[b] See text for explanation.

Table 2
F_{max} and EER obtained with the first 5, 10, and 50 AdaBoost-selected features, and with all (several hundred) extracted features

# features	F_{max} %				EER %			
	5	10	50	All	5	10	50	All
Cars	92.3	90.8	92.5	80.6	91.4	89.9	91.4	79.9
Horses	69.4	81.7	83.0	71.6	69.4	81.1	82.2	70.0
Apple logos	83.5	84.1	89.2	84.3	81.8	84.1	84.1	79.5
Bottles	59.3	61.4	72.2	68.6	58.2	60.0	70.9	65.5
Mugs	64.6	75.6	83.5	74.4	62.1	71.2	81.8	69.7

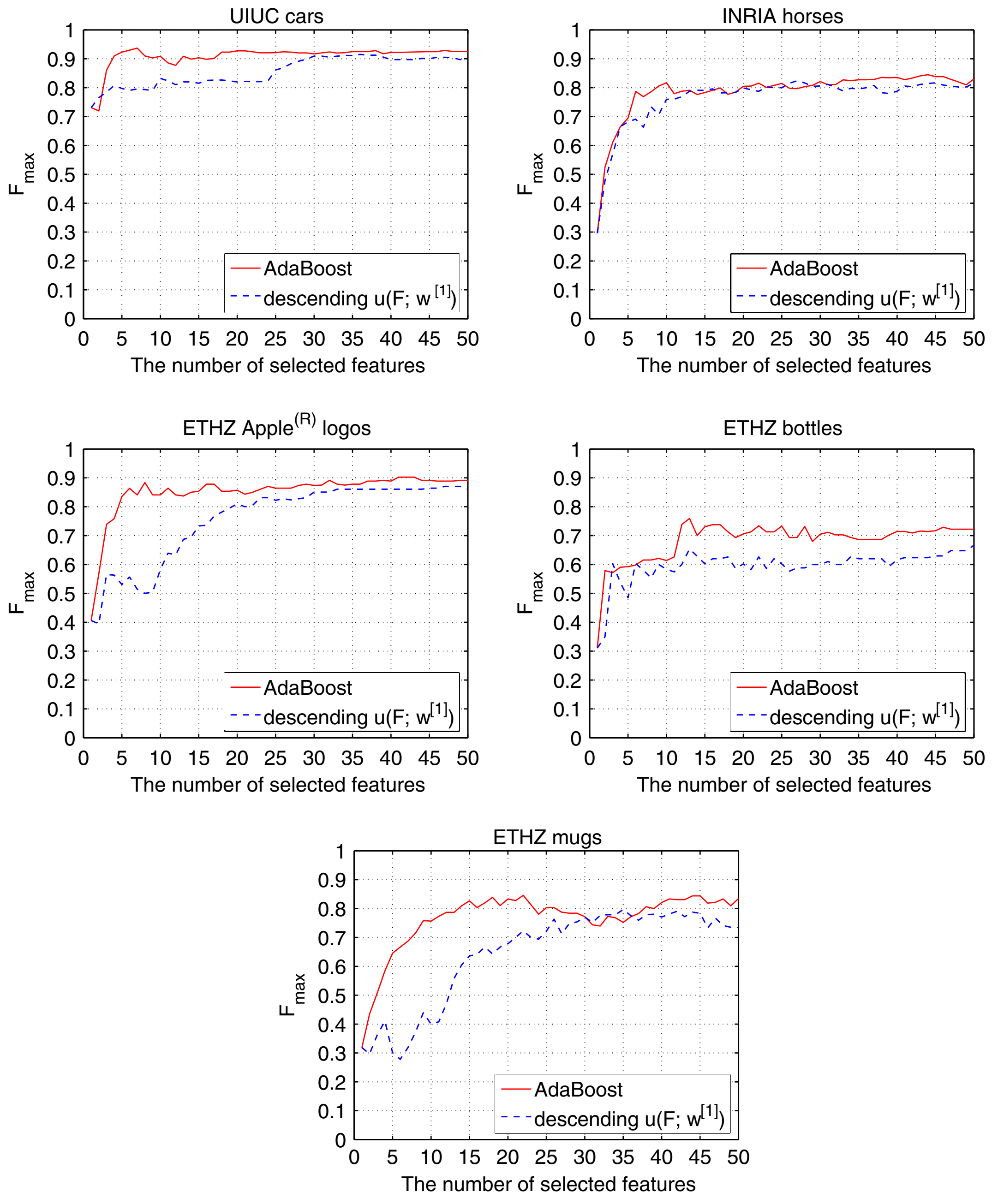


Fig. 3. The influence of the number of selected features on the detection accuracy. The AdaBoost selection algorithm is compared with a procedure that selects features in descending order of $u(F; \mathbf{w}^{[1]})$.

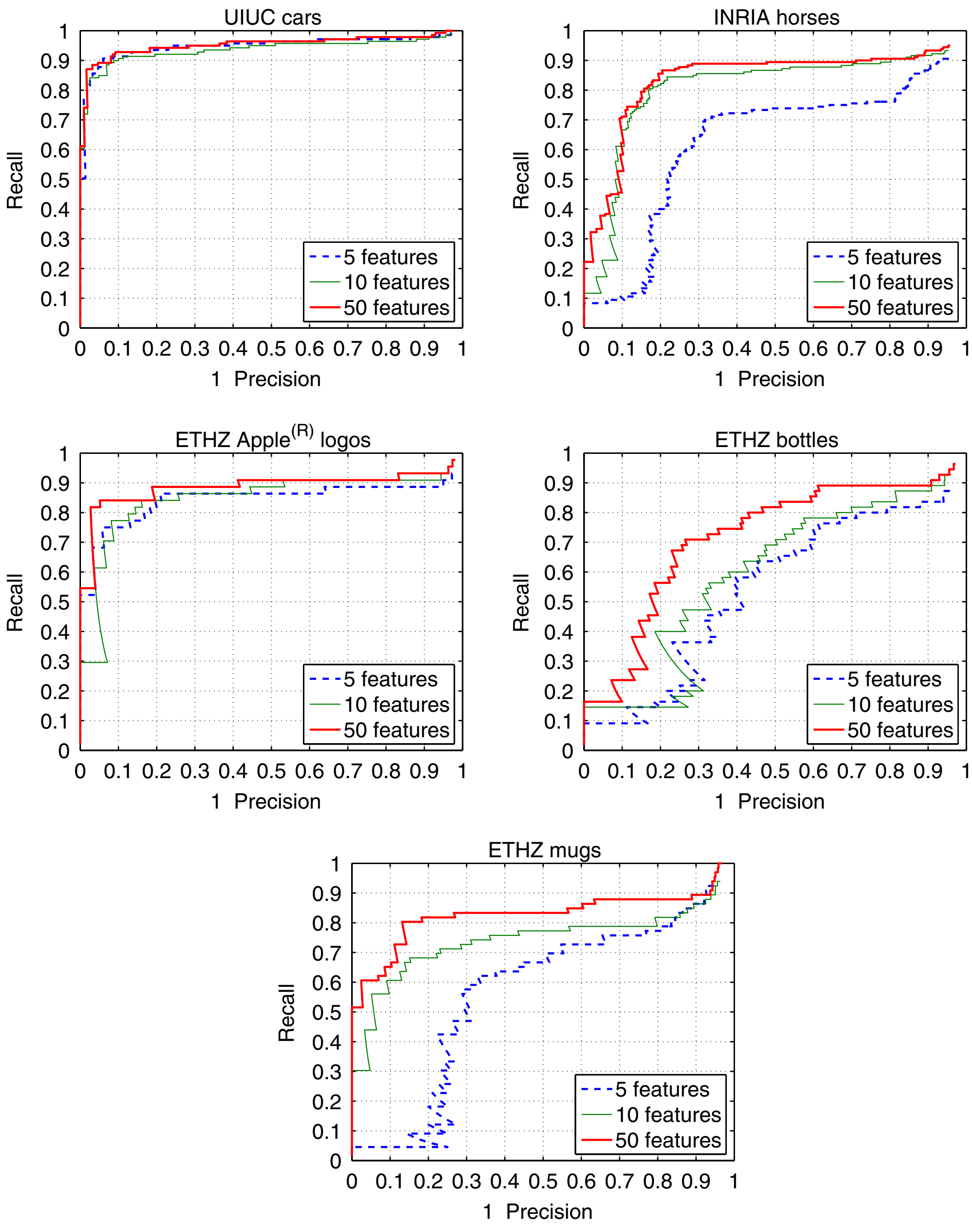


Fig. 4. Recall–precision curves obtained using the first 5, 10, and 50 AdaBoost-selected features.

5.1. Training stage implementation details

The training stage makes use of two image sets: the first consists only of positive images and is used in the feature extraction and ISM learning procedures, while the second consists of both positive and negative images and is used in the feature selection procedure. In all setups, the two training sets were completely separated from the test set, as they were acquired from different sources. In the UIUC setup, the TU Darmstadt dataset[5] (100 images) served as the first training set, and the UIUC training dataset (550 positive, 500 negative images) served as the second. In all other setups, the first training set also played the role of the positive subset of the second set.

The Weizmann horse dataset[6] (Borenstein and Ullman, 2002) was used in the training stage for the INRIA setup. The training images for the ETHZ setups, as well as the negative image set for the INRIA setup, were acquired from various Web sources, including Google Image Search.

Table 1 lists the number of training examples and the average training object size for each detection setup.

5 http://www.pascal-network.org/challenges/VOC/databases.html.

6 http://www.msri.org/people/members/eranb.

5.2. Test stage implementation details

Every test image was analyzed at every scale within a given setup-specific range. The employed scale ranges (the last column of Table 1) typically encompassed 11–12 levels in the scale-space pyramid, or 2.5–2.75 octaves, considering that each octave comprised four levels.

5.3. Evaluation criteria

When evaluated on a given test image, the detector returns a set of hypothesized bounding boxes, each of which is characterized by its center $[(x_H, y_H)]$, width (w_H), and height (h_H). Let x_T, y_T, w_T, and h_T denote the center and the size of the true bounding box of an object. A hypothesis is counted as correct if the following conditions are satisfied:

- The distance between the true and the hypothesized object center, relative to the true object size, is below a specified threshold. Formally,

$$\sqrt{\left(\frac{2(x_H - x_T)}{w_T}\right)^2 + \left(\frac{2(y_H - y_T)}{h_T}\right)^2} \leqslant 0.5. \tag{12}$$

- The true bounding box covers at least 50% of the area of the hypothesized bounding box.
- The hypothesized bounding box covers at least 50% of the area of the true bounding box.

The above evaluation criterion, introduced by Leibe (2004), was adopted for all detection setups except UIUC. Several authors have experimented with the UIUC dataset, so we decided to compare our results with theirs. To make this possible, the criterion proposed by Agarwal et al. (2004) had to be used. According to this criterion, a hypothesis is considered correct if

$$\frac{(x_T - x_H)^2}{\alpha_h^2} + \frac{(y_T - y_H)^2}{\alpha_w^2} + \frac{(w_T - w_H)^2}{\alpha_\sigma^2} \leqslant 1, \tag{13}$$

where $\alpha_h = 0.25h_T$ and $\alpha_w = \alpha_\sigma = 0.25w_T$.

Following Leibe (2004) and Agarwal et al. (2004), we incorporated an additional rule into both criteria: If multiple hypotheses correctly predict the same object, only the strongest one is accepted as correct, while all others are counted as false positives (i.e., incorrect hypotheses).

5.4. Detection accuracy measure

As argued by Agarwal et al. (2004), the most suitable measure for evaluating detection accuracy on a given test set is a *recall–precision curve* (RPC). Let $\mathscr{H}$ denote the set of all detection hypotheses, correct and incorrect, gathered for all images in the given test set. *Recall* and *precision* are defined as follows:

$$\mathrm{Recall}(\tau) = \frac{|\mathrm{TP}(\tau)|}{\#\ \mathrm{objects}}, \tag{14}$$

$$\mathrm{Precision}(\tau) = \frac{|\mathrm{TP}(\tau)|}{|\mathrm{TP}(\tau)| + |\mathrm{FP}(\tau)|}. \tag{15}$$

In the above definition, #objects denotes the total number of objects of interest in the entire test set. $\mathrm{TP}(\tau)$ and $\mathrm{FP}(\tau)$ denote the sets of true positives and false positives, respectively, whose strength is at least τ:

$$\mathrm{TP}(\tau) = \{H \in \mathscr{H} \,|\, (H \text{ is correct}) \wedge (s(H) \geqslant \tau)\}, \tag{16}$$

$$\mathrm{FP}(\tau) = \{H \in \mathscr{H} \,|\, (H \text{ is incorrect}) \wedge (s(H) \geqslant \tau)\}. \tag{17}$$

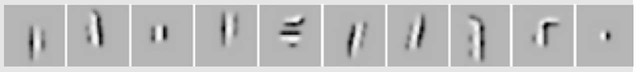
As τ increases, the number of hypotheses in $\mathrm{TP}(\tau)$ and $\mathrm{FP}(\tau)$ decreases.

RPC is a parametric curve (τ being the parameter) that visualizes the relationship between $(1 - \mathrm{Precision}(\tau))$ and $\mathrm{Recall}(\tau)$ as τ increases from the strength of the weakest hypothesis in $\mathscr{H}$ to the strength of the strongest hypothesis in $\mathscr{H}$. An RPC may be characterized by its *maximum F-measure*:

$$F_{\max} = \max_\tau F(\tau) = \max_\tau \frac{2 \cdot \mathrm{Recall}(\tau) \cdot \mathrm{Precision}(\tau)}{\mathrm{Recall}(\tau) + \mathrm{Precision}(\tau)}. \tag{18}$$

Another commonly employed measure for evaluating RPCs is *equal-error rate* (EER), which is defined as recall at the point where recall and precision are equal.

Table 3
The first ten selected features for each setup. The selection algorithm was presented with all features extracted at layer 3. On average, the extracted set comprised 843 features per detection setup

Dataset	The first 10 AdaBoost-selected features
Cars	
Horses	
Apple logos	
Bottles	
Mugs	

A more detailed discussion on RPCs is offered in Agarwal et al. (2004).

5.5. Experimental results and discussion

Fig. 3 visualizes the relationship between the number of selected features and the detection accuracy for individual setups. The figure also compares the developed AdaBoost selection scheme with a strategy that simply selects features in descending order of evaluation scores $u(F; \mathbf{w}^{[1]})$. Features selected by AdaBoost prove to be significantly more appropriate for the experimental object detection tasks than features selected according to $u(F; \mathbf{w}^{[1]})$. As noted in Section 2.3, AdaBoost involves a weight-adjustment mechanism that discourages redundancy in the selected set.

The obtained results provide affirmative evidence for the devised AdaBoost selection scheme. In the UIUC setup, $F_{\max} = 91.0\%$ and EER$= 90.7\%$ when cars are detected using just four AdaBoost-selected features. With as few as seven features, the detection accuracy reaches its maximum ($F_{\max} = 93.7\%$, EER $= 92.1\%$). Table 2 shows the dependence of both $F_{\max}$ and EER on the number of selected features. If the selection stage is bypassed and all extracted features are used in the test stage (the 'All' column in Table 2), the detection accuracy drops, which further demonstrates the significance of feature selection.

Fig. 4 displays recall–precision curves for individual detection setups.

The first 10 AdaBoost-selected features for each category are displayed in Table 3. Despite their apparent simplicity, these features collectively possess sufficient strength to bring about a relatively high detection accuracy. As shown in Table 4, our results for the UIUC dataset are comparable to results reported by other authors.

Table 4
Detection results for the UIUC car dataset: a comparison between our results (obtained using 5, 10, or 50 AdaBoost-selected features) and results reported by other authors

Method	$F_{\max}$ (%)	EER (%)
Ours, 5 features	92.3	91.4
Ours, 10 features	90.8	89.9
Ours, 50 features	92.5	91.4
Agarwal et al. (2004)	44.0	39.6
Fritz et al. (2005)	Not given	87.8
Mutch and Lowe (2006)	Not given	90.6

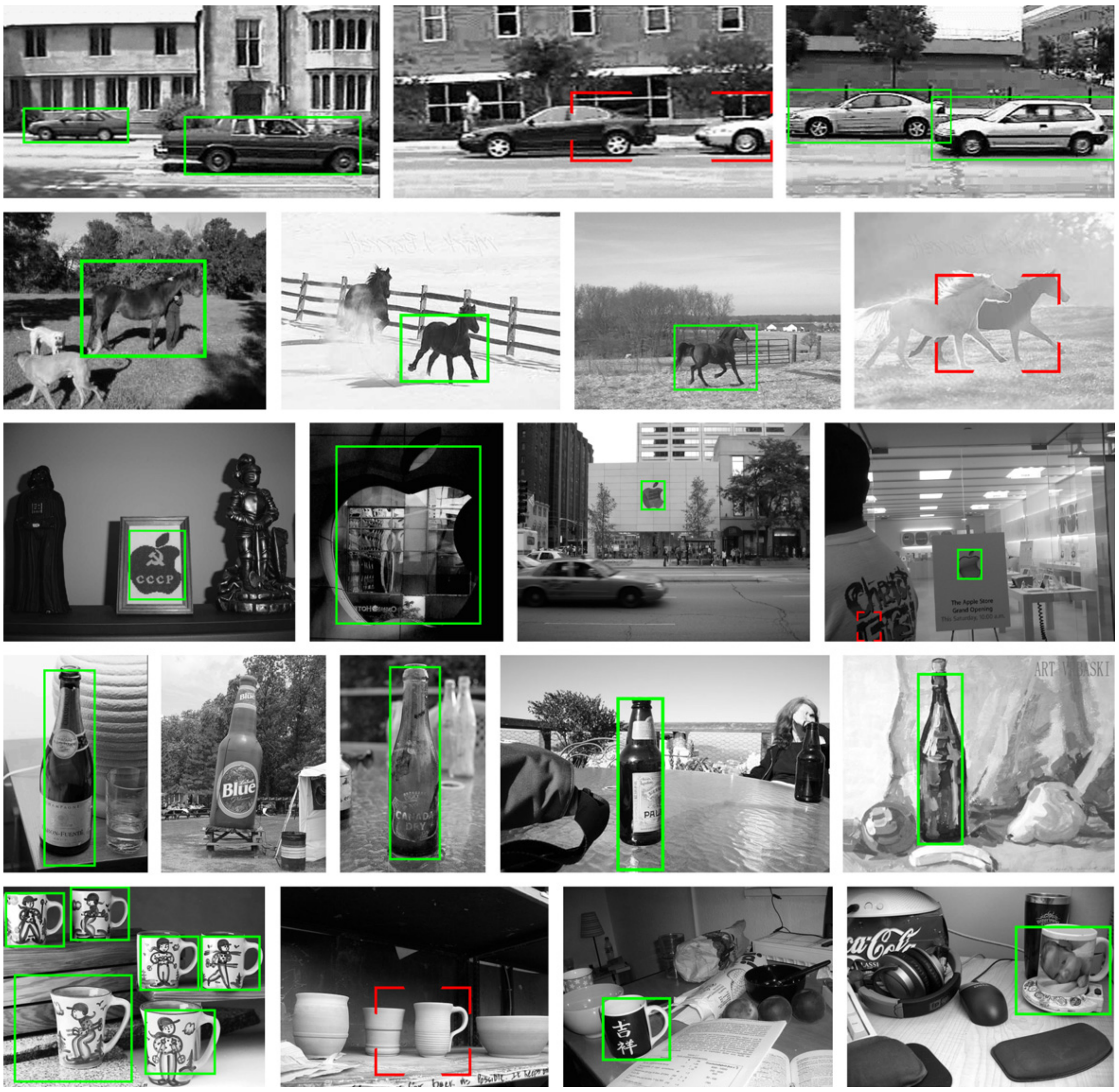


Fig. 5. Some examples of detections for all five datasets using the first 10 AdaBoost-selected features. The detection threshold was set to the value that led to the maximum F-measure (Eq. (18)). A lower threshold would result in fewer missing detections and more false positives, and vice versa.

The AdaBoost curves in Fig. 3 typically grow very rapidly at the beginning, but then they start to stagnate or fluctuate. We surmise that detection accuracy could be further improved only by increasing the complexity of extracted features. Simple features are able to distinguish a few essential patterns, which, as we demonstrated, suffices for some detection tasks. However, relying exclusively on simple features imposes an accuracy barrier that cannot be crossed merely by enlarging the population of selected features. An effective selection strategy quickly discovers the features that capture the most essential patterns, but more complex patterns can be distinguished only by using more complex features. Since the specificity of features grows with increasing complexity, a comparatively large number of complex features might have to be selected to adequately cover the essential patterns. If the selection procedure were provided with more complex input features, the curves in Fig. 3 would probably grow more slowly but would eventually surpass the curves obtained for the employed features.

Fig. 5 provides some concrete examples of correct and incorrect detections. False positives commonly arise in highly cluttered areas, but sometimes they are due to inherent properties of the detection model. The second image in the first row is a rare example of a failure due to the attenuation algorithm. The correct hypothesis had initially been slightly weaker than the highlighted false positive. Both hypotheses had received much of their score from the rear wheel of the left car. As a consequence of both facts, the algorithm significantly attenuated the correct hypothesis but left the false positive intact.

Feature selection considerably affects the temporal efficiency of the detection algorithm in the test stage. Using the first 5 AdaBoost-selected features, the algorithm consumes 3.9 s on average to process a single UIUC test image at all seven scales. With 50 features, the consumed amount of time rises to 16.6 s per image. With all 920 features, the detection algorithm requires 117.8 s on average per image. All experiments were run on an Intel Core 2 Duo machine using a Matlab implementation.

6. Conclusion

We devised a novel feature evaluation function for object detection and embedded it into an AdaBoost-based selection procedure. The feature selection module was combined with the feature extraction method of Fidler et al. (2006) and the object detection method of Leibe et al. (2004).

The effectiveness of the developed selection strategy was experimentally confirmed. In three out of five detection setups, as few as six simple features suffice to achieve a fairly high level of detection accuracy. Whereas the selection algorithm undoubtedly plays a significant role, the overall success of the method should also be attributed to the feature extraction and object detection modules.

Currently, the feature selection module can be combined only with a single-category detection algorithm. Our future research might thus be directed toward adapting the selection technique for the multi-category detection domain, where the objective is to predict the category of each hypothesized object in addition to its location and size.

Another potentially interesting research issue is automatic determination of the optimal *number* of selected features. A viable approach to this problem is to design a criterion that evaluates the trade-off between the given number of selected features and the corresponding detection accuracy on a training image set.

References

Agarwal, S., Awan, A., Roth, D., 2004. Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Machine Intell. 26 (11), 1475–1490.

Borenstein, E., Ullman, S., 2002. Class-specific, top-down segmentation. In: European Conference on Computer Vision, pp. 109–124.

Cover, T., 1974. The best two independent measurements are not the two best. IEEE Trans. Systems Man Cybernet. 4 (1), 116–117.

Dorkó, G., Schmid, C., 2003. Selection of scale-invariant parts for object class recognition. In: Internat. Conf. on Comput. Vision, pp. 634–640.

Epshtein, B., Ullman, S., 2005. Feature hierarchies for object classification. In: Internat. Conf. on Comput. Vision, pp. 220–227.

Ferrari, V., Fevrier, L., Jurie, F., Schmid, C., 2006. Groups of adjacent contour segments for object detection. Tech. Rep., INRIA Rhône-Alpes.

Fidler, S., Leonardis, A., 2007. Towards scalable representations of object categories: Learning a hierarchy of parts. In: IEEE Conf. on Comput. Vision and Pattern Recognition.

Fidler, S., Berginc, G., Leonardis, A., 2006. Hierarchical statistical learning of generic parts of object structure. In: IEEE Conf. on Comput. Vision and Pattern Recognition, pp. 182–189.

Freund, Y., Schapire, R., 1997. A decision-theoretic generalization of online learning. Comput. System Sci. 55 (1), 119–139.

Fritz, M., Leibe, B., Caputo, B., Schiele, B., 2005. Integrating representative and discriminant models for object category detection. In: Internat. Conf. on Comput. Vision. pp. 1363–1370.

Leibe, B., 2004. Interleaved object categorization and segmentation. Ph.D. Thesis, ETH Zürich.

Leibe, B., Leonardis, A., Schiele, B., 2004. Combined object categorization and segmentation with an implicit shape model. In: ECCV'04 Workshop on Statist. Learning in Comput. Vision, pp. 17–32.

Leonardis, A., Gupta, A., Bajcsy, R., 1995. Segmentation of range images as the search for geometric parametric models. Internat. J. Comput. Vision 14 (3), 253–277.

Li, F., Košecká, J., Wechsler, H., 2006. Strangeness based feature selection for part based recognition. In: IEEE Conference on Computer Vision and Pattern Recognition Workshop. p. 22.

Mikolajczyk, K., Leibe, B., Schiele, B., 2006. Multiple object class detection with a generative model. In: IEEE Conf. on Comput. Vision and Pattern Recognition, pp. 26–33.

Mutch, J., Lowe, D., 2006. Multiclass object recognition with sparse, localized features. In: IEEE Conf. on Comput. Vision and Pattern Recognition, pp. 11–18.

Opelt, A., Pinz, A., Zisserman, A., 2006. A boundary-fragment-model for object detection. In: Eur. Conf. on Comput. Vision, pp. 575–588.

Peng, H., Long, F., Ding, C., 2005. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Machine Intell. 27 (8), 1226–1238.

Shotton, J., Blake, A., Cipolla, R., 2005. Contour-based learning for object detection. In: Internat. Conf. on Comput. Vision, pp. 503–510.

Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Schiele, B., Gool, L.V., 2006. Towards multi-view object class detection. In: IEEE Conf. on Comput. Vision and Pattern Recognition, pp. 1589–1596.

Torralba, A., Murphy, K., Freeman, W., 2004. Sharing features: efficient boosting procedures for multiclass object detection. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 762–769.

Viola, P., Jones, M., 2004. Robust real-time face detection. Internat. J. Comput. Vision 57 (2), 137–154.

Zhang, W., Yu, B., Zelinsky, G., Samaras, D., 2005. Object class recognition using multiple layer boosting with heterogeneous features. In: IEEE Conf. on Comput. Vision and Pattern Recognition, pp. 323–330.